Robustness Analysis of Multi-Agent Patrolling Strategies using Reinforcement Learning

Fabrice Lauri and Abderrafiaa Koukam

IRTES-SeT
Rue Thiery-Mieg, 90010 Belfort
{fabrice.lauri,abder.koukam}@utbm.fr

Abstract. Patrolling an environment involves a team of agents whose goal usually consists in continuously visiting the most relevant areas as fast as possible. In this paper, we follow up on the work by Santana *et al.* who formulated this problem in terms of a reinforcement learning problem, where agents individually learn an MDP using Q-Learning to patrol their environment. We propose another definition of the state space and of the reward function associated with the MDP of an agent. Experimental evaluation shows that our approach substantially improves the previous RL method in some situations (graph topology and number of agents). Moreover, it is observed that such an RL approach is able to cope efficiently with most of the situations caused by the removal of agents during a patrolling simulation.

1 Introduction

The multi-agent patrolling problem has been rigourously addressed only recently [1–7]. In these works, many patrolling strategies have been devised and experimentally validated using common evaluation criteria [1]. They are based on different approaches, ranging from heuristic laws enabling agents to better choose the next node to visit [1], negotiation mechanisms [2], reinforcement learning techniques [3], techniques based on graph theory [4] to techniques based on ACO [5–7]. Most of these solutions yield good empirical results on different graphs constituted from less than fifty nodes and one hundred edges. Nevertheless, none of these solutions have been evaluated in terms of robustness. In some applications though, one might also want to know how the performances of one of these solutions are influenced by an *online* change in the size of the population of the individuals (or agents) involved in a patrol.

In this paper, we first propose an improvement of the learning agents' architecture presented in [3], by characterizing more precisely the MDP employed by an agent. Moreover, we experimentally show that a reinforcement learning based approach can be efficiently applied to the multi-agent patrolling problem for dealing with both an increase in the graph complexity and an online addition or removal of agents. We chose the reinforcement learning framework for studying the robustness of the patrolling problem for two main reasons. On the one hand, a machine learning approach can theoretically cope with any graph

topology and any agents' set, so that a larger range of situations can be considered. On the other hand, we assume that all the agents are located at the same node at the initial time. Under this condition, the patrolling task starts with a preliminary phase where agents spread out in the graph. This step cannot be handled by the most efficient techniques based on Single Cycle [4, 2], which limits agents to be located at different nodes.

The remainder of this paper is organized as follows. Section 2 describes the commonly used framework of a patrolling problem and gives an overview of the related works. Section 3 reviews the fundamental concepts of a reinforcement learner. Section 4 proposes a specification of the patrolling problem in terms of a reinforcement learning problem. Experimental results are shown in section 5. Finally, concluding remarks and future research issues are given in section 6.

2 Problem Definition

The patrolling problem is usually specified formally as follows [1, 4, 3]. The environment to patrol is reduced to a graph $G = (V, E)$, V representing the strategically relevant areas and E the safe ways of movement or communication between them. A cost c_{ij}, associated with each edge (i, j), measures the time required to go from node i to node j. Let be r agents bound to visit at regular intervals the areas defined in the graph G. Each agent is located at one of the nodes of V at the initial time. Solving the patrolling problem consists of elaborating a multi-agent graph coverage strategy π. Such a strategy must optimize a given quality criterion. $\pi = \{\pi_1 \cdots \pi_r\}$ is made up of the r individual strategies π_i of each agent i. An individual strategy π_i is defined such that $\pi_i : \mathbb{N} \rightarrow V$, $\pi_i(j)$ denoting the j-th node visited by the agent i.
Intuitively, a relevant patrolling strategy is one that minimizes, for each node, the time span between two visits to the same node. Several criteria have been devised in [1] in order to evaluate the quality of a multi-agent patrolling strategy after T time steps (or *cycles*) of simulation. All of them are based on the notion of *instantaneous node idleness* (INI). The INI $I_t(i)$ of a node i at time t is the number of time steps this node remained unvisited. By convention, at the initial instant, $I_0(i) = 0$, $\forall i = 1, 2, \cdots, |\mathcal{V}|$. At a given instant t, GI_t is the *instantaneous average graph idleness* (IGI). Similarly the *instantaneous worst graph idleness* WI_t is the highest INI encountered since t time steps of simulation. A multi-agent patrolling strategy π can be evaluated after T cycles of simulation using either the *average idleness* criterion AI_π or the *worst idleness* WI_π. The average idleness denotes the mean of the IGI over the T simulation cycles, whereas the *worst idleness* is the highest INI observed during the T-time steps of the simulation. As emphasized by [4], the optimal strategy π is the one that minimizes the worst idleness, as $WI_\pi \geq AI_\pi$ for any strategy π.

3 Reinforcement Learning Framework

Reinforcement learning typically deals with problems where one or several agents interact with their environment to learn to perform a task. At each time step, an

agent is able to (1) perceive the state of its environment, (2) carry out an action which modifies the environment state and (3) obtain an immediate reward depending on the action it just performed. After several thousands of trials, such an agent learns a policy π, which tells him what to do in every situation [8]. A reinforcement learning problem involving one agent is usually defined in terms of a *Markov Decision Process* (MDP). Several extensions of an MDP, such as MMDP [9] or DEC-MDP [10] have been proposed to deal with the problem of coordination in multi-agent systems, but these solutions are intractable when the number of agents is high. Indeed, they use joint actions whose number exponentially increases with the number of individual actions and agents involved: if there are n agents, each of which can perform a actions, then the size of the joint action space is a^n. To alleviate this problem, many approaches [11–13, 3] consider RL agents as *independent learners*. Independent learners ignore the actions and rewards of the other agents, and learn their policy using their own MDP. Although these approaches are no longer assumed to find a globally optimal solution, they still yield satisfactory results in practice. For this reason, as the patrolling problem may involve a lot of agents, we will focus in this paper on the case where agents employ an MDP learned with Q-Learning to perform its task.

4 Learning to Patrol using Reinforcements

One of the most difficult tasks when designing a patrolling agent's MDP is the definition of its state space. As each of our agents uses partial information to find a globally optimal solution to the patrolling problem, the more features are incorporated in a state, the more precise the solution can be. On the other hand, it is well known that the size of the state space grows exponentially with the number of features. Defining the state space of the MDPs is thus a trade-off between their computational complexity and the global solution approximation they yield. In [3], the learning agents' architecture which obtained the best results was *Gray-Box Learner Agent* (GBLA) when using the idleness of the next reached node as the immediate reward. This architecture incorporates into an agent's MDP some information characterizing its environment vicinity and enables each of them to communicate its intention about its next action. As this architecture constituted the first attempt to formulate the patrolling problem in a reinforcement learning framework, it is unfortunately not perfect. In the next sections, we will discuss the drawbacks of this architecture and see how its definition can be refined.

4.1 Identifying the dark side of MDP

Considering that d stands for the graph degree and $|\mathcal{V}|$ is the number of nodes of the graph, the state space S in GBLA was made up of the following components : (1) the node where the agent is ($|\mathcal{V}|$ possible values) (2) the edge from which it came (d possible values) (3) the neighbor node which has the highest (worst) idleness (d possible values) (4) the neighbor node which has the lowest idleness

(d possible values) and (5) the list of the adjacent nodes which are intended to be visited by other agents (2^d possible values). The cardinality of the action set was equal to the graph degree d, each action enabling an agent to reach an adjacent node. As emphasized previously, the size of the state space grows exponentially with the number of features : with this MDP definition, the total number of states $|S| = |\mathcal{V}| \times d^3 \times 2^d$ and the total number of actions $|A| = d$. Learning several MDPs (one for each patrolling agent) with Q-Learning can therefore become rapidly intractable when a lot of agents try to patrol in a graph of high degree and a great number of nodes. For instance, with a graph with a degree of 7 and constituted by 50 nodes (the graph called *map A* in the previous works), each MDP needs theoretically to store more than 15 million scalars (used by the Q-table).

4.2 Numbering the valid states using the graph topology

Yet, among the $|\mathcal{V}| \times d^3 \times 2^d$ states, a lot of them will never be visited by an agent. For instance, let us consider the five-node graph $G = (V, E)$, where $V = \{1, 2, 3, 4, 5\}$ and $E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{4, 5\}\}$. Here, the MDP of an agent patrolling this graph will be potentially made up of $|S| = 5 \times 3^3 \times 2^3 = 1080$ states. But some states will never been encountered by the agent, such as the ones in which it is in node 3 and came from a node reached when performing action 2 in node 3. In fact, the number of states that will be effectively visited by an agent (the *valid states*) can be computed precisely from the graph topology. Assuming that d_i is the degree of node i, the number of valid states induced by this MDP definition is equal to $|S| = \sum_{i=1}^{|\mathcal{V}|} d_i^3 \times 2^{d_i}$. Using this formula, the size of the state space associated to this graph is reduced to $|S| = 252$, which is about one quarter of the size of the initial state space for this graph topology. Using this simple principle, since the number of states is now reduced to a sum over the total number of graph nodes, graphs with a greater amount of nodes can be dealt with. This reasoning can be pushed further by considering that only d_i actions at node i can be performed. The number of valid indices (s, a) can thus similarly be reduced when a particular scalar must be accessed through the Q-table. For instance, when dealing with map A, no more than 1.6 million scalars would need to be stored, which is here nearly one tenth of the size of the initial state space. In order to take into account only the valid states and thus avoid allocating too much memory, a variant of Q-Learning will be used to learn each MDP. This version of Q-Learning maintains an ordered list of the states that have been visited at least once. When $Q(s, a)$ must be accessed from a state s and an action a (through the Q-table), a search of the corresponding valid state s' is initiated in the *already visited states* list. If state s does not exist in it, it is added at the end. Else, its order number s' in the list is used as the first part of the index for the Q-table.

4.3 Adding more local information in a state representation

We just saw that the topology of the graph to be patrolled can considerably reduce the size of the state space, so that a better characterization of the envi-

ronment vicinity and of the information required to coordinate the agents' action can possibly be incorporated into a state vector. Indeed, it seems to us that the MDP defined in GBLA was incomplete. It is incomplete because the third and fourth features of a state (the neighboring node which has the highest idleness and the neighboring node which has the lowest idleness) do not precisely inform the agent about its environment vicinity if more than two edges are connected to a node. In order to enable an agent to decide which is the best action to execute in a given state using the most relevant information, we redefined the MDP associated to an agent in terms of its state space and reward function. Firstly, we suggest to represent the following features on the state : (1) the node where the agent is ($|\mathcal{V}|$ possible values), (2) the edge from which an agent came from (d_i possible values), (3) an ordered list of the adjacent nodes from node i, sorted according to their idleness ($d_i!$ possible values), (4) the list of the adjacent nodes from node i which are intended to be visited by other agents (2^{d_i} possible values). The number of valid states induced by this state space is equal to $|S| = \sum_{i=1}^{|\mathcal{V}|} d_i \times d_i! \times 2^{d_i}$. Secondly, the immediate reward given to an agent is equal to zero if the reached node was bound to be visited by other agents, else it is equal to the idleness of the reached node. This new model will be called Extended-GBLA in the remainder of this article.

5 Experimental Results

Multi-agent patrolling strategies were trained on the six different graph topologies commonly used by the community, with populations of 2 to 15 agents.

To obtain strategies as robust as possible, their training were divided into trials. At each trial, graph statistics (the node idlenesses and the average graph idleness) was set to zero, all the agents were placed at the same starting node and they learned to patrol during several iterations. The starting node changed from one trial to the other. Patrolling strategies were trained using either GBLA or Extended-GBLA. Thus, a total of 120 patrolling strategies (10×6 for each RL method) were trained. Preliminary experiments were conducted to determine the learning parameters, such as the number of trials, the number of iterations per trial, the learning rate α, the discount factor γ and the exploration probability ϵ. The agents' MDPs were trained using 1000 trials, 10000 iterations per trial, $\alpha = 0.9$, $\gamma = 0.9$ and $\epsilon = 0.1$. Two classes of experiments were carried out to assess the robustness of our multi-agent patrolling strategies. The first ones were conducted in order to know whether the patrolling strategies trained with GBLA or with Extended-GBLA are still efficient when the node where all agents start to patrol is changed. The second experiments measure the capacity of the patrolling agents to adapt on situations where some agents broke down.

5.1 Comparison of GBLA and Extended-GBLA

Figure 1 presents the average graph idleness obtained after a multi-agent patrolling simulation using strategies trained with GBLA and Extended-GBLA.

Each trained patrolling strategy was evaluated 20 times by changing the starting node of agents and by using 50000 cycles of simulation. Thus, subsequent results represent the average graph idleness over the 20 runs. Confidence intervals indicated on figures were computed using a risk of 5%. One can already see

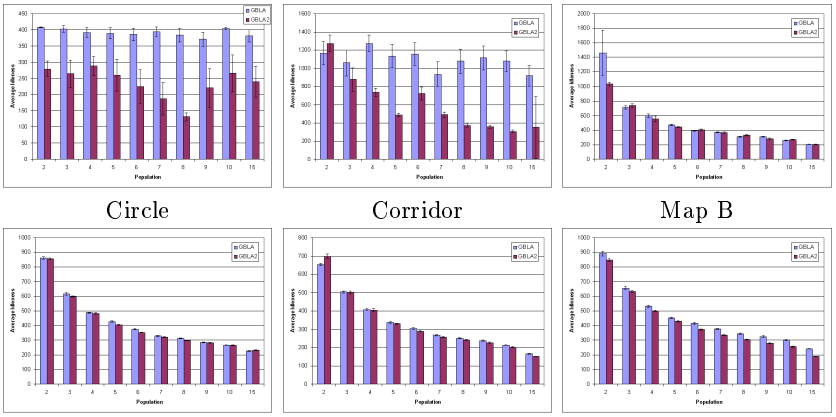


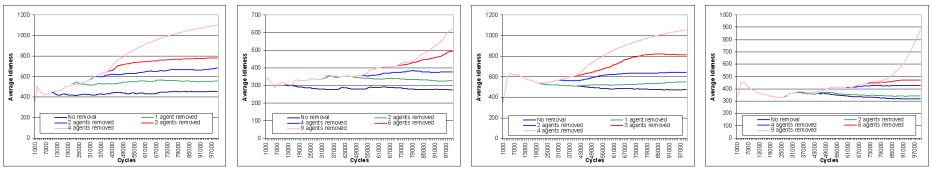
Fig. 1. Comparison results between GBLA and Extended-GBLA

that for the Map A, the Map B, the Grid Map and the Island Map, agents have learned to coordinate their actions, since the average graph idleness decreases when the number of agents increases. Despite their lowest degree, patrolling on the Corridor-shaped graph and on the Circle-shaped graph seems to be more complicated. Both RL methods give equivalent performance for the Map B, the Grid Map and the Island Map. For the other graphs, Extended-GBLA is significantly better than GBLA. By observing the agents' behavior on our simulator, we classified patrolling agents into two different classes. The first class is composed of agents that are responsible of only one region of the graph: they patrol only nodes of that region during the whole simulation. The second class is made up of agents that cross from one region to another one, especially to visit the node which links several regions, thus avoiding to decrease performances. These behaviors were only observed on the four more complex graphs (Map A, Map B, Island and Grid) with both GBLA and Extended-GBLA, and on the two lowest complex graphs (Corridor and Circle) only with Extended-GBLA. With GBLA, all the agents follow the same politics on the latter graphs : they all cross the graphs in the same direction and at the same time. This explains why the average graph idleness does not decrease when the agents' population grows. It is not the case using Extended-GBLA. We explain this phenomenon by considering that with Extended-GBLA, agents are informed of the utility to go to a given node through the reward function: if one agent intends to visit a node, the other agents will not want to visit it as it will give a zero reward. Hence, the definition

of the reward function of Extended-GBLA enables agents to better coordinate their action. From this point of view, we can say that the information added by Extended-GBLA to the state space of MDPs used by agents (that is the ordered list of the adjacent nodes sorted according to their idleness) seems to have less influence than reward functions do on the performance of the patrolling strategies.

5.2 Robustness of Extended-GBLA

Figure 2 shows the influence of the removal of agents on the average graph idleness obtained after a 100000-cycle patrolling simulation using multi-agent strategies trained with Extended-GBLA. These experiments were carried out on the Corridor map and on the Map B with 5 and 10 agents. One such experiment consisted of choosing how many agents will be removed (or will fall into a breakdown) during the simulation, considering that the first agent will be removed after the 10000th cycle and that each subsequent agent will be removed every 10000 cycles. Results show that for both graphs, strategies are no more efficient



Map B, 5 agents | Map B, 10 agents | Corridor, 5 agents | Corridor, 10 agents

Fig. 2. Robustness results of Extended-GBLA

when only one agent remains to patrol. Indeed, as an agent has only a local representation of its environment when it uses an MDP, it often forgets to visit some nodes for a while, thus decreasing performance. This forgetting behavior can also be observed when more than one agent remains to patrol (for instance on Map B with 10 agents where 6 agents are removed). In this case, it is due to the specialization of some agents that learned to patrol only in a given area of the graph. In the other cases, when a sufficient number of agents are patrolling, one can observe that agents are able to rapidly adapt to the new situations.

6 Concluding Remarks and Future Works

We have proposed in this article a novel definition of MDPs used by agents to learn individually how to patrol in a graph. The RL algorithm Q-Learning was used to give agents the capability to select the best actions to carry out in a given situation in a dynamic environment (that is where agents continuously move). We have experimentally shown that our RL method Extended-GBLA significantly outperforms in several graph topologies the approach proposed by

Santana *et al.* [3]. We believe this improvement is mainly due to the redefinition of the reward function, which allows agents to better coordinate their actions. Moreover, results evaluating the robustness of Extended-GBLA reveal that the patrolling strategies trained with this method are still efficient when some agents are removed during a patrolling simulation. However, the trained strategies were unable to cope adequately with situations where only a few agents remain to patrol. To tackle this problem, an adaptation phase seems to be required to allow the remaining patrolling agents to face new situations caused by a removal of a lot of agents. Future research directions include the use of an undiscounted RL method, such as R-Learning [14], to better characterize this problem by optimizing the average reward.

References

1. Machado, A., Ramalho, G., *al*: Multi-Agent Patrolling : an Empirical Analysis of Alternatives Architectures. In: 3rd International Workshop on Multi-Agent Based Simulation. (2002) 155–170
2. Almeida, A., Ramalho, G., *al*: Recent Advances on Multi-Agent Patrolling. In: 17th Brazilian Symposium on Artificial Intelligence. (2004) 474–483
3. Santana, H., Ramalho, G., *al*: Multi-Agent Patrolling with Reinforcement Learning. In: 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems. (2004) 1122–1129
4. Chevaleyre, Y.: Theoretical Analysis of the Multi-Agent Patrolling Problem. In: International Joint Conference on Intelligent Agent Technology. (2004) 302–308
5. Lauri, F., Charpillet, F.: Ant Colony Optimization applied to the Multi-Agent Patrolling Problem. In: IEEE Swarm Intelligence Symposium. (2006)
6. Lauri, F., Koukam, A.: A Two-Step Evolutionary and ACO Approach for Solving the Multi-Agent Patrolling Problem. In: IEEE World Congress on Computational Intelligence. (2008)
7. Lauri, F., Koukam, A.: Hybrid ACO/EA Algorithms applied to the Multi-Agent Patrolling Problem. In: IEEE World Congress on Computational Intelligence. (2014)
8. Sutton, R., Barto, A. In: Reinforcement Learning : An Introduction. Cambridge, MA (1998)
9. Boutilier, C.: Sequential Optimality and Coordination in Multi-Agent Systems. In: 16th International Joint Conference on Artificial Intelligence. (1999) 478–485
10. Bernstein, D., Zilberstein, S., Immerman, N.: The Complexity of Decentralized Control of Markov Decision Processes. In: 16th Conference on Uncertainty in Artificial Intelligence. (2000) 32–37
11. Sen, S., Sekaran, M., Hale, J.: Learning to coordinate without sharing information. In: 12th National Conference on Artificial Intelligence. (1994) 426–431
12. Schneider, J., Wong, W., Moore, A., Riedmiller, M.: Distributed value functions. In: 16th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA (1999) 371–378
13. Wolpert, D., Wheeler, K., Tumer, K.: General Principles of Learning-based Multi-Agent Systems. In: 3rd International Conference on Autonomous Agents (Agents'99). (1999) 77–83
14. Schwartz, A.: A reinforcement learning method for maximizing undiscounted rewards. In: 10th International Conference on Machine Learning. (1993) 298–305